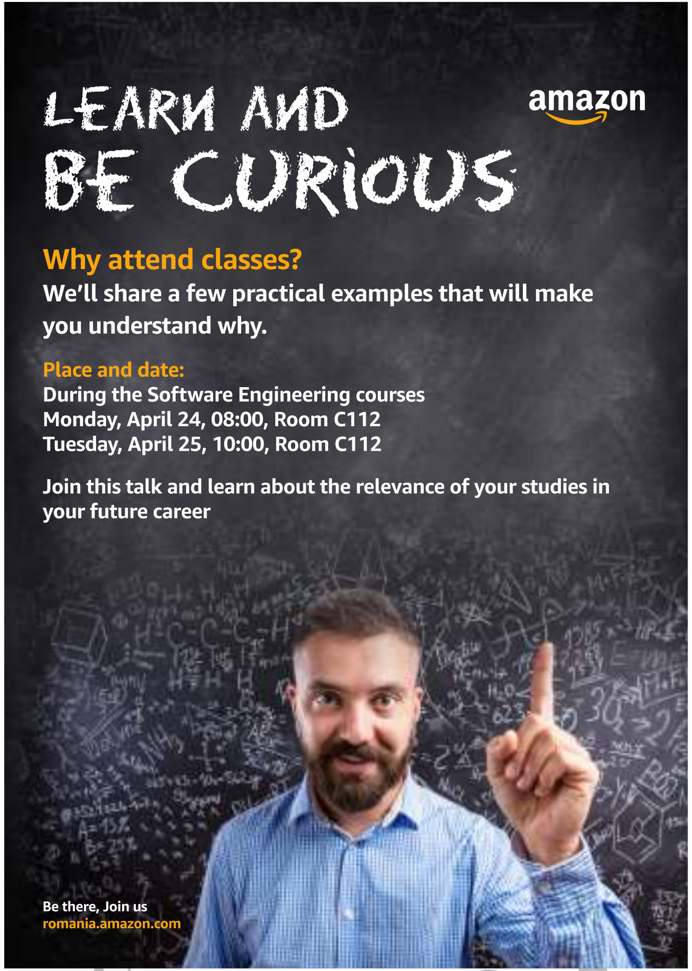
- **What** is **Learn&Be Curious?**
- **Why** we are here
- **What** we’ll talk about



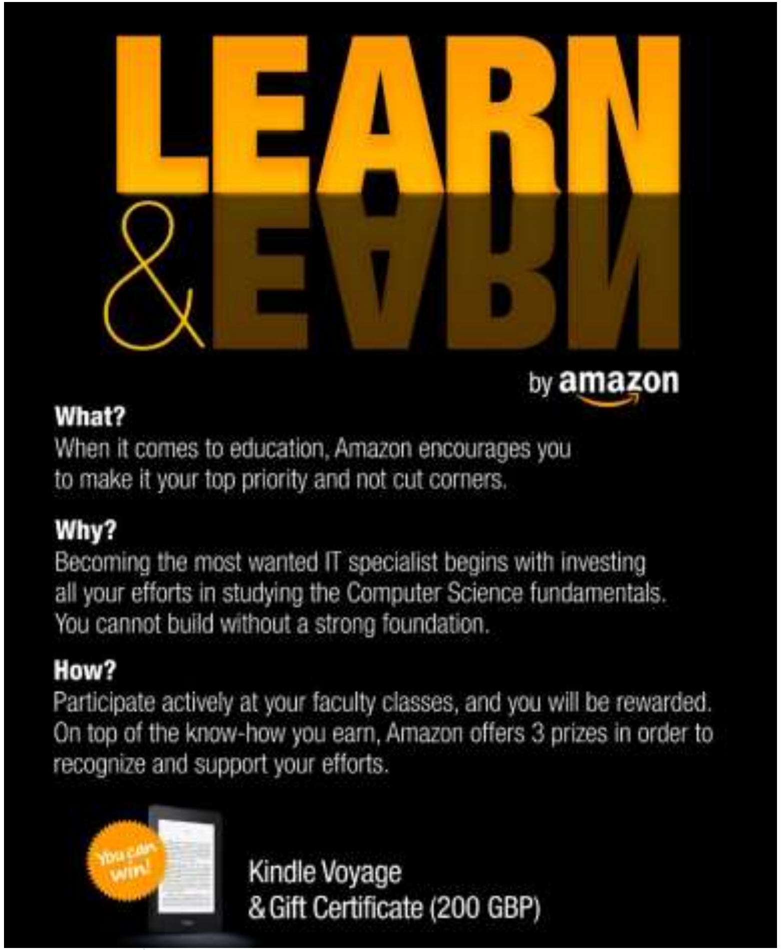
Why attend classes?
We’ll share a few practical examples that will make you understand why.

Place and date:
During the Software Engineering courses
Monday, April 24, 08:00, Room C112
Tuesday, April 25, 10:00, Room C112

Join this talk and learn about the relevance of your studies in your future career

Be there, Join us
romania.amazon.com

- **What** is **Learn&Earn**
- **Why** award prizes?


LEARN
&
EARN
by amazon
You can win!

What?
When it comes to education, Amazon encourages you to make it your top priority and not cut corners.

Why?
Becoming the most wanted IT specialist begins with investing all your efforts in studying the Computer Science fundamentals. You cannot build without a strong foundation.

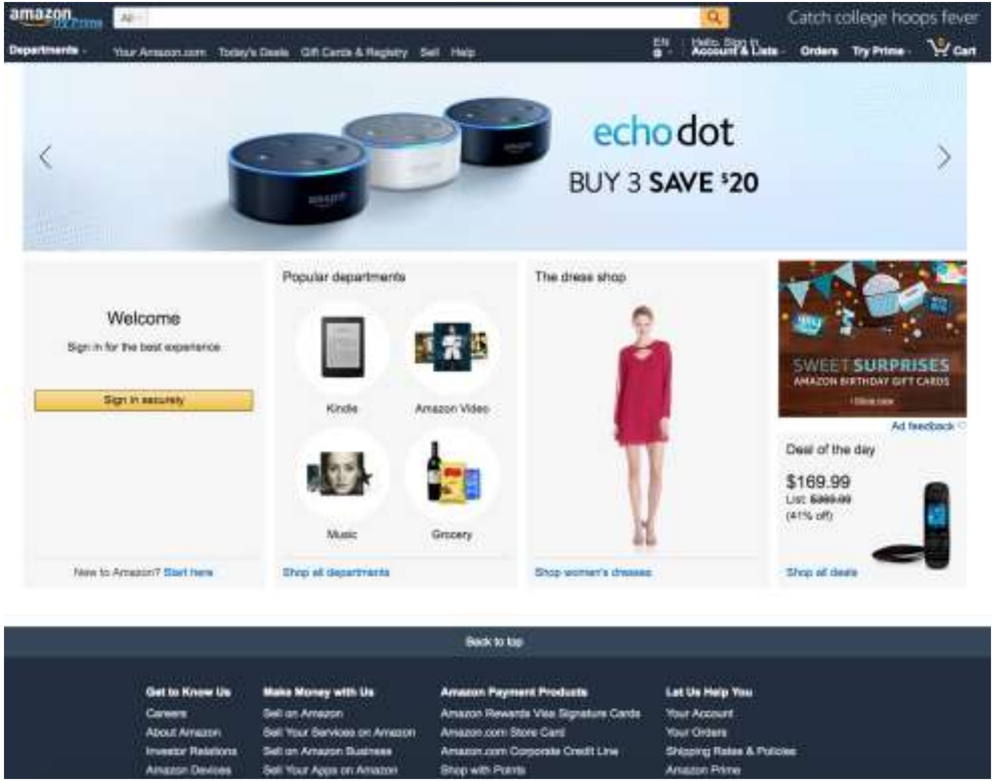
How?
Participate actively at your faculty classes, and you will be rewarded. On top of the know-how you earn, Amazon offers 3 prizes in order to recognize and support your efforts.

Kindle Voyage
& Gift Certificate (200 GBP)

Small or world changing,
at the heart of any software is engineering.

What is amazon?

Online retail



Catch college hoops fever

echo dot

BUY 3 SAVE $20

Welcome

Sign in for the best experience

New to Amazon? Start here

Deal of the day

$169.99

List: ~~$289.99~~

(41% off)

What is amazon?

Online Retail

Robotics

What is amazon?

Online Retail

Robotics

Prime Air Delivery

What is amazon?

Online Retail

Robotics

Prime Air Delivery

AWS

What is amazon?

- Online Retail
- Robotics
- Prime Air Delivery
- AWS
- **Kindle**

What is amazon?

Online Retail

Robotics

Prime Air Delivery

AWS

Kindle

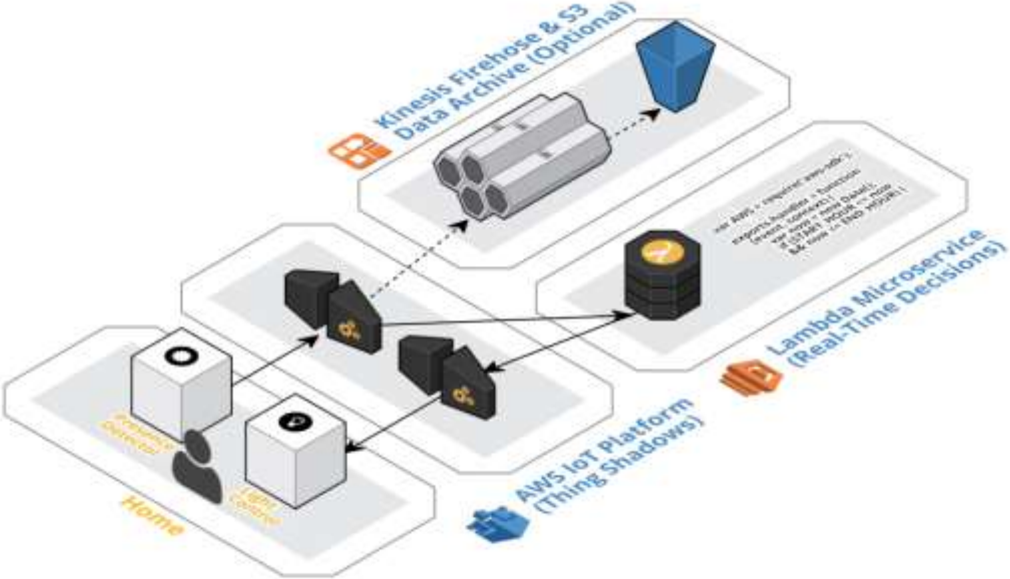
Amazon Go

What is amazon?

- Online Retail
- Robotics
- Prime Air Delivery
- AWS
- Kindle
- Amazon Go
- **Echo**

What is amazon?

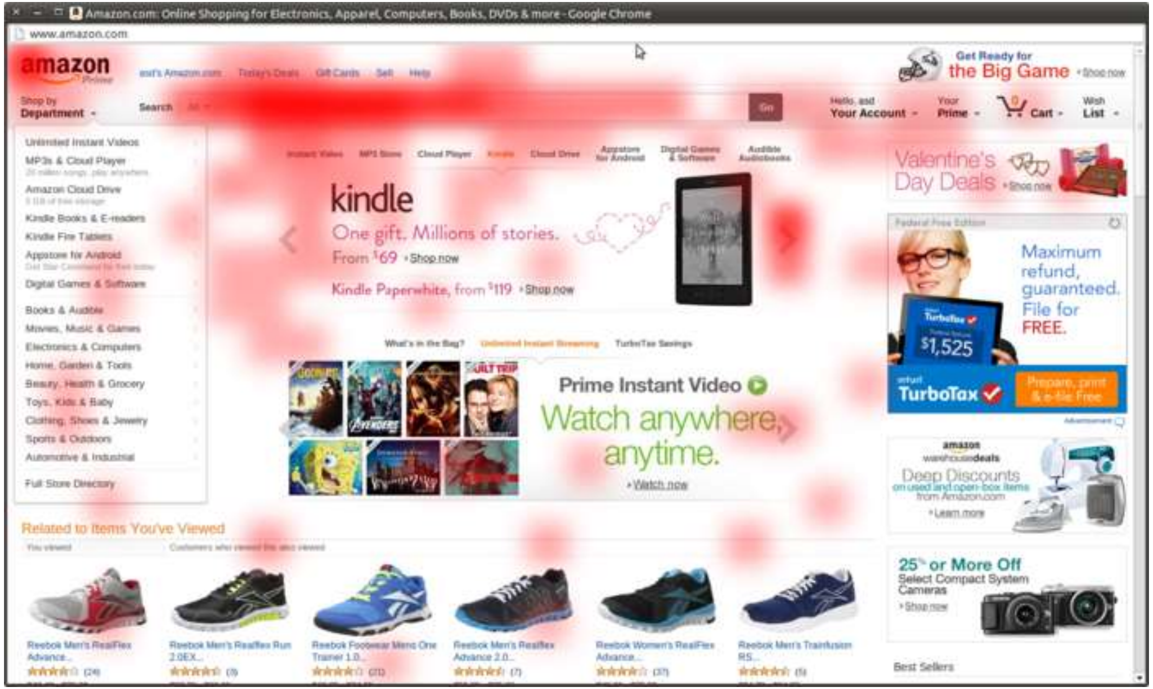
- Online Retail
- Robotics
- Prime Air Delivery
- AWS
- Kindle
- Amazon Go
- Echo
- **Internet of Things**



“Put the customer first. **Invent**. And be patient.”

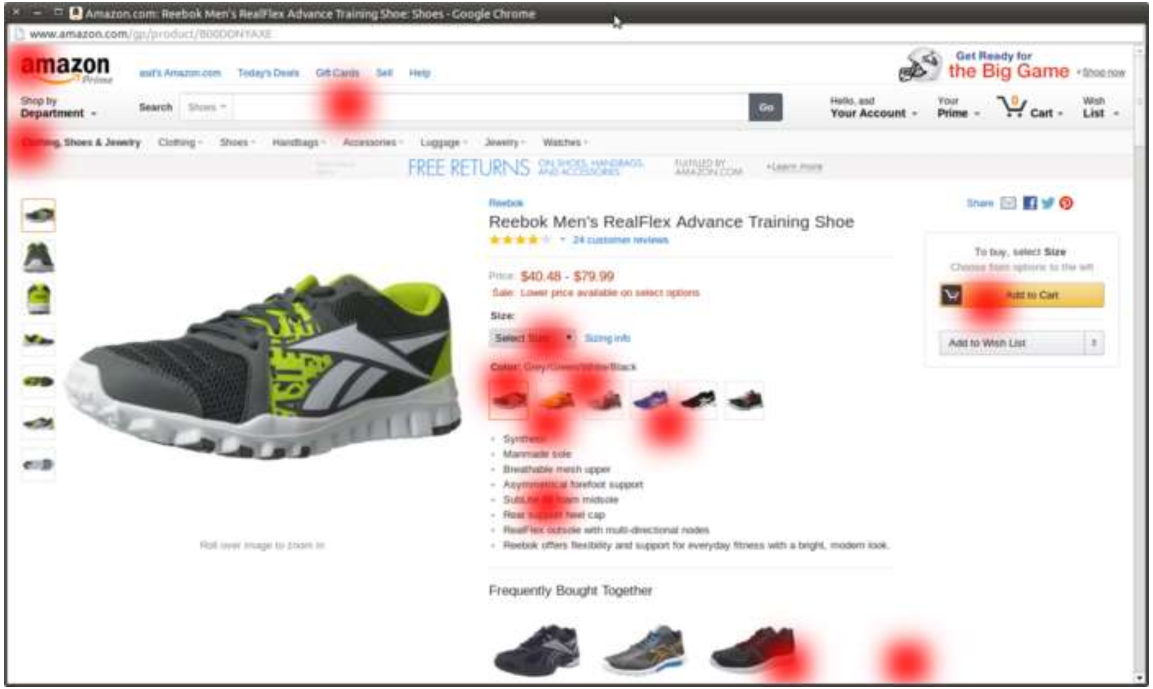
Optimizing Amazon.com

- How can we understand what the user's journey is on a page?
- In-page analytics. *Everyone does it!*



Optimizing Amazon.com

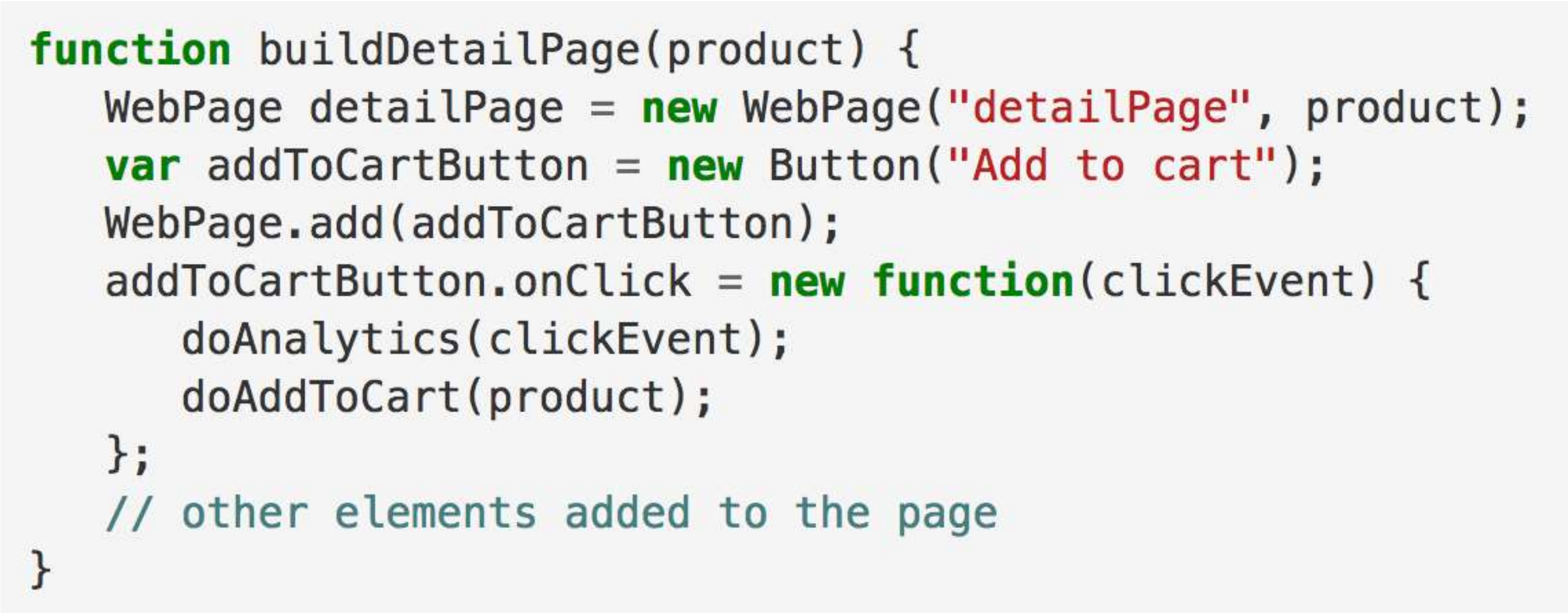
- Measure, measure, measure!
- In the wild, not in the lab.
- Improve on the basis of real data.


Amazon.com: Reebok Men's RealFlex Advance Training Shoe: Shoes - Google Chrome
www.amazon.com/gp/product/B00DONYAXE
amazon Prime
asd's Amazon.com
Today's Deals
Gift Cards
Sell
Help
Get Ready for the Big Game
› Shop now
Shop by Department
Search
Shoes
Go
Hello, asd Your Account
Your Prime
0 Cart
Wish List
Clothing, Shoes & Jewelry
Clothing
Shoes
Handbags
Accessories
Luggage
Jewelry
Watches
FREE RETURNS ON SHOES, HANDBAGS, AND ACCESSORIES
FULFILLED BY AMAZON.COM
› Learn more
Reebok
Reebok Men's RealFlex Advance Training Shoe
24 customer reviews
Price: $40.48 - $79.99
Sale: Lower price available on select options
Size:
Select Size
Sizing info
Color: Grey/Green/Yellow/Black
Synthetic
Manmade sole
Breathable mesh upper
Asymmetrical forefoot support
SubLite fill foam midsole
Rear support heel cap
RealFlex outsole with multi-directional nodes
Reebok offers flexibility and support for everyday fitness with a bright, modern look.
Roll over image to zoom in
Frequently Bought Together
Share
To buy, select Size
Choose from options to the left
Add to Cart
Add to Wish List

Analytics hooks

- A web page is composed of web elements.
- A user interacts with web elements.
- Interaction triggers more than one unit of work to be run.

Analytics hooks

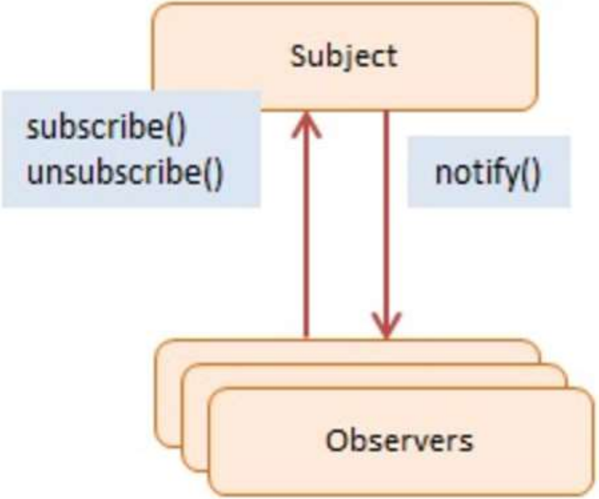


```
function buildDetailPage(product) {
    WebPage detailPage = new WebPage("detailPage", product);
    var addToCartButton = new Button("Add to cart");
    WebPage.add(addToCartButton);
    addToCartButton.onClick = new function(clickEvent) {
        doAnalytics(clickEvent);
        doAddToCart(product);
    };
    // other elements added to the page
}
```

Can we decouple the analytics from the actual add to cart invocation?

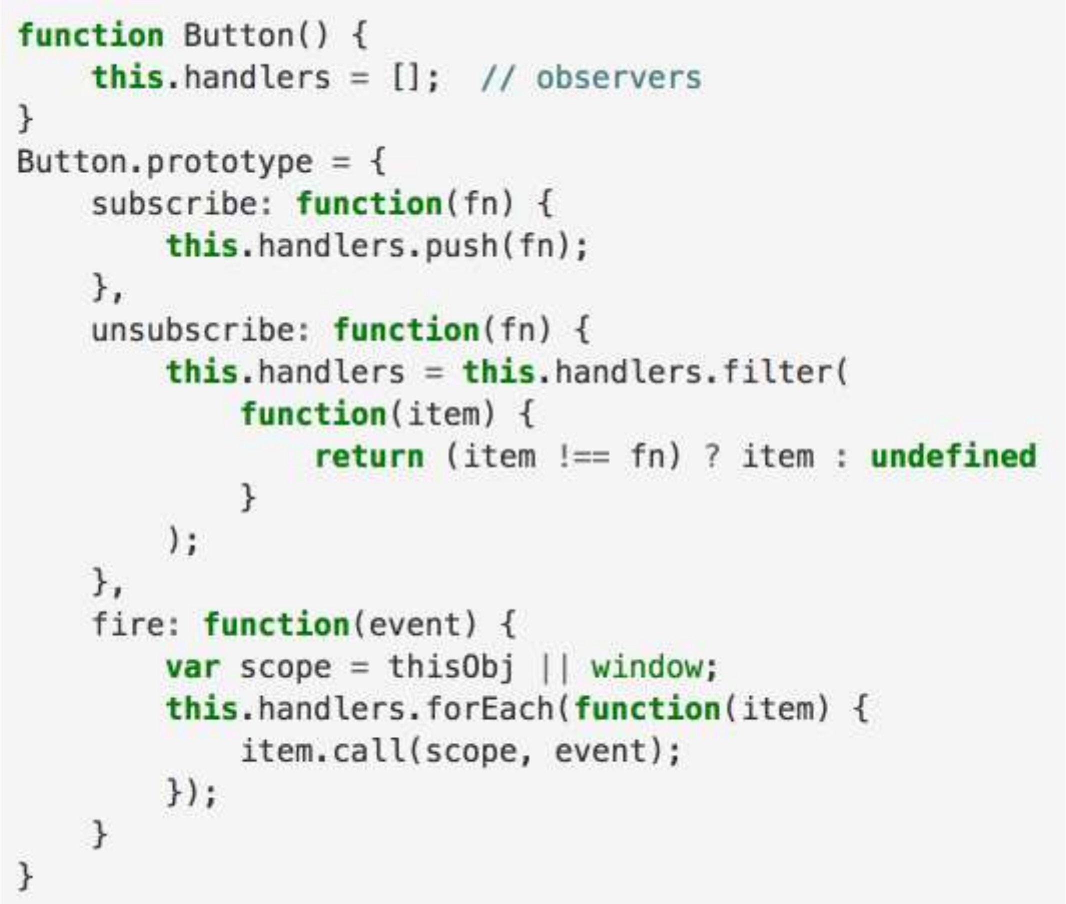
Observer

- **Problem:** A large monolithic design does not scale well as new graphing or monitoring requirements are levied.
- **Definition**: Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.
- **Web apps** ⬄ writing many **event handlers.**
- **Event handlers** = functions that will be notified when a certain event fires.



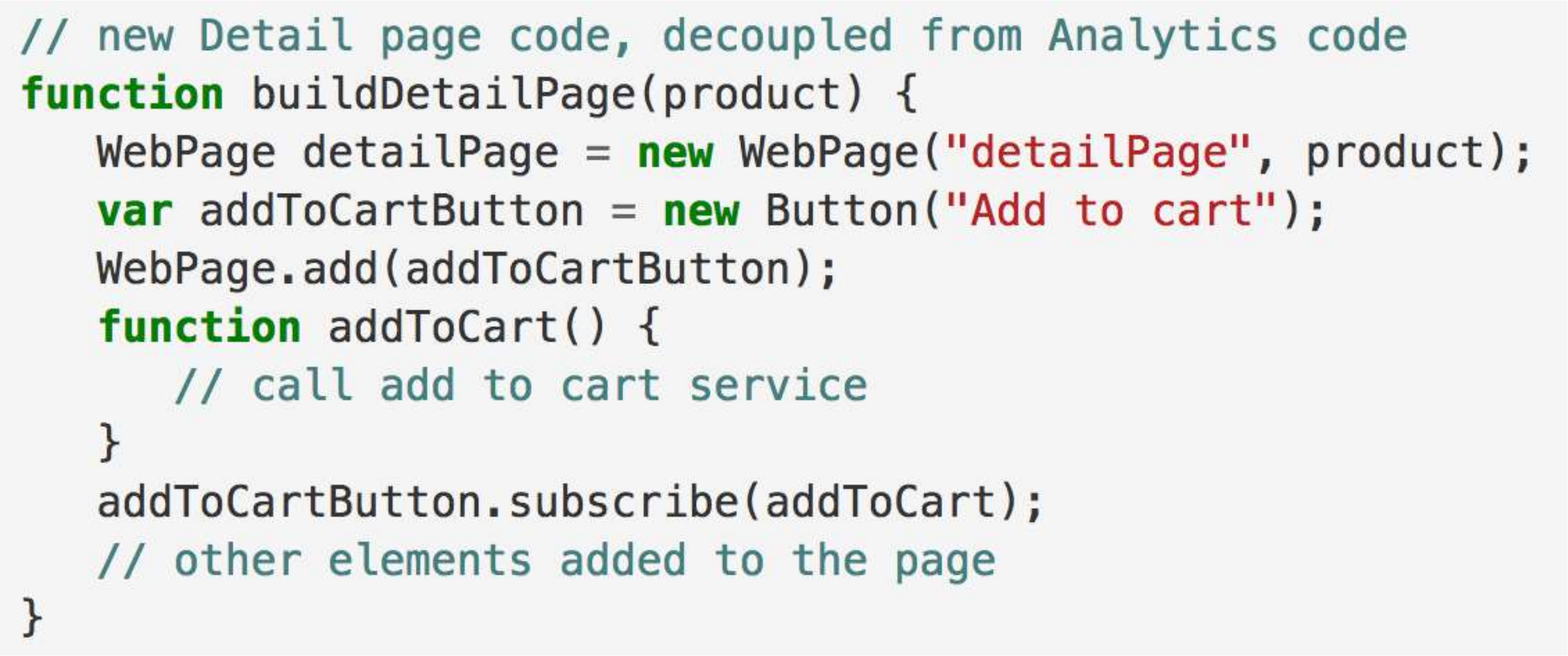
http://www.dofactory.com/javascript/observer-design-pattern

Observer



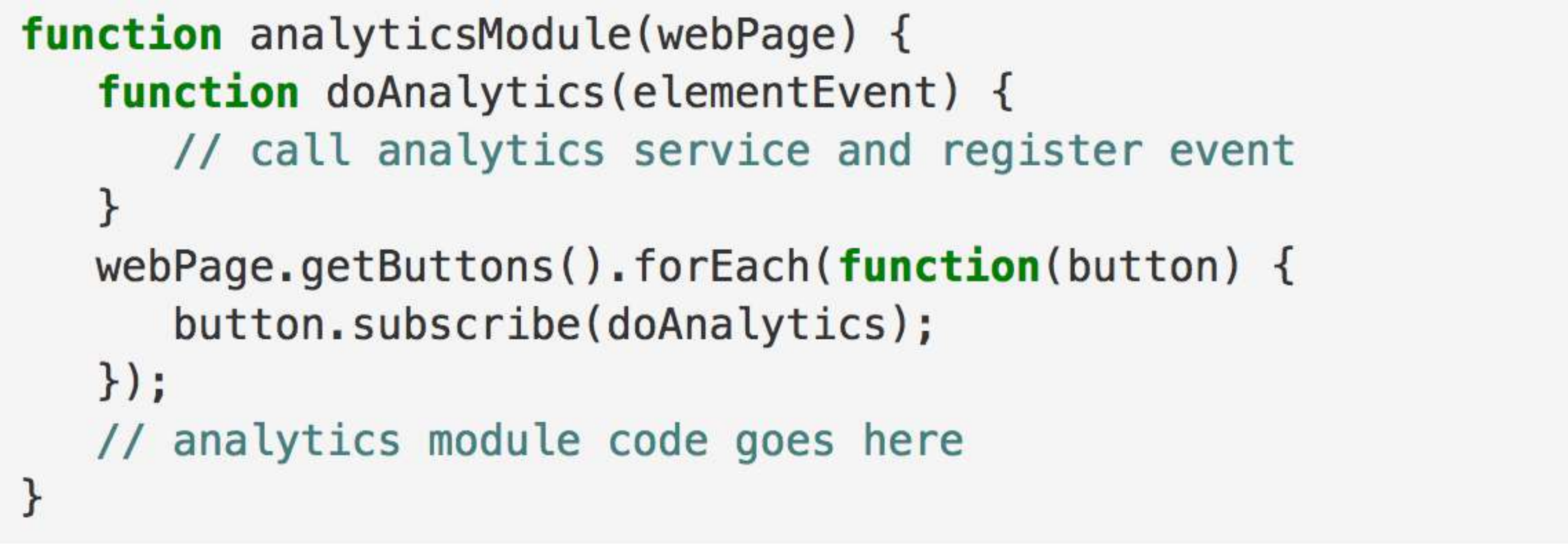
```
function Button() {
    this.handlers = [];  // observers
}
Button.prototype = {
    subscribe: function(fn) {
        this.handlers.push(fn);
    },
    unsubscribe: function(fn) {
        this.handlers = this.handlers.filter(
            function(item) {
                return (item !== fn) ? item : undefined
            }
        );
    },
    fire: function(event) {
        var scope = thisObj || window;
        this.handlers.forEach(function(item) {
            item.call(scope, event);
        });
    }
}
```

Observer



```
// new Detail page code, decoupled from Analytics code
function buildDetailPage(product) {
   WebPage detailPage = new WebPage("detailPage", product);
   var addToCartButton = new Button("Add to cart");
   WebPage.add(addToCartButton);
   function addToCart() {
      // call add to cart service
   }
   addToCartButton.subscribe(addToCart);
   // other elements added to the page
}
```

Observer



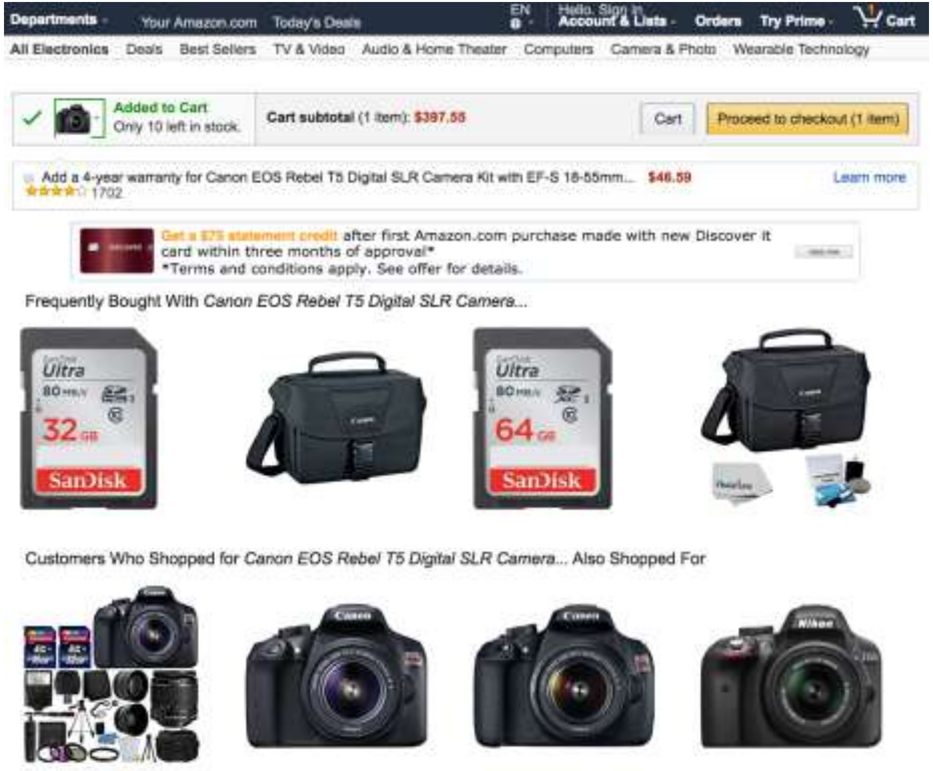
```
function analyticsModule(webPage) {
    function doAnalytics(elementEvent) {
        // call analytics service and register event
    }
    webPage.getButtons().forEach(function(button) {
        button.subscribe(doAnalytics);
    });
    // analytics module code goes here
}
```

Observer - benefits

- Decoupling of subject and observers: the subject doesn’t need to know about how many or which observers will be interested in changes.
- Open-closed software: we only need to write new observer code, not modify existing one.

Shopping engagement

- How are these recommendations served?
- What if we want new types of recommendations to be displayed?



Added to Cart
Only 10 left in stock.

Cart subtotal (1 item): $397.55

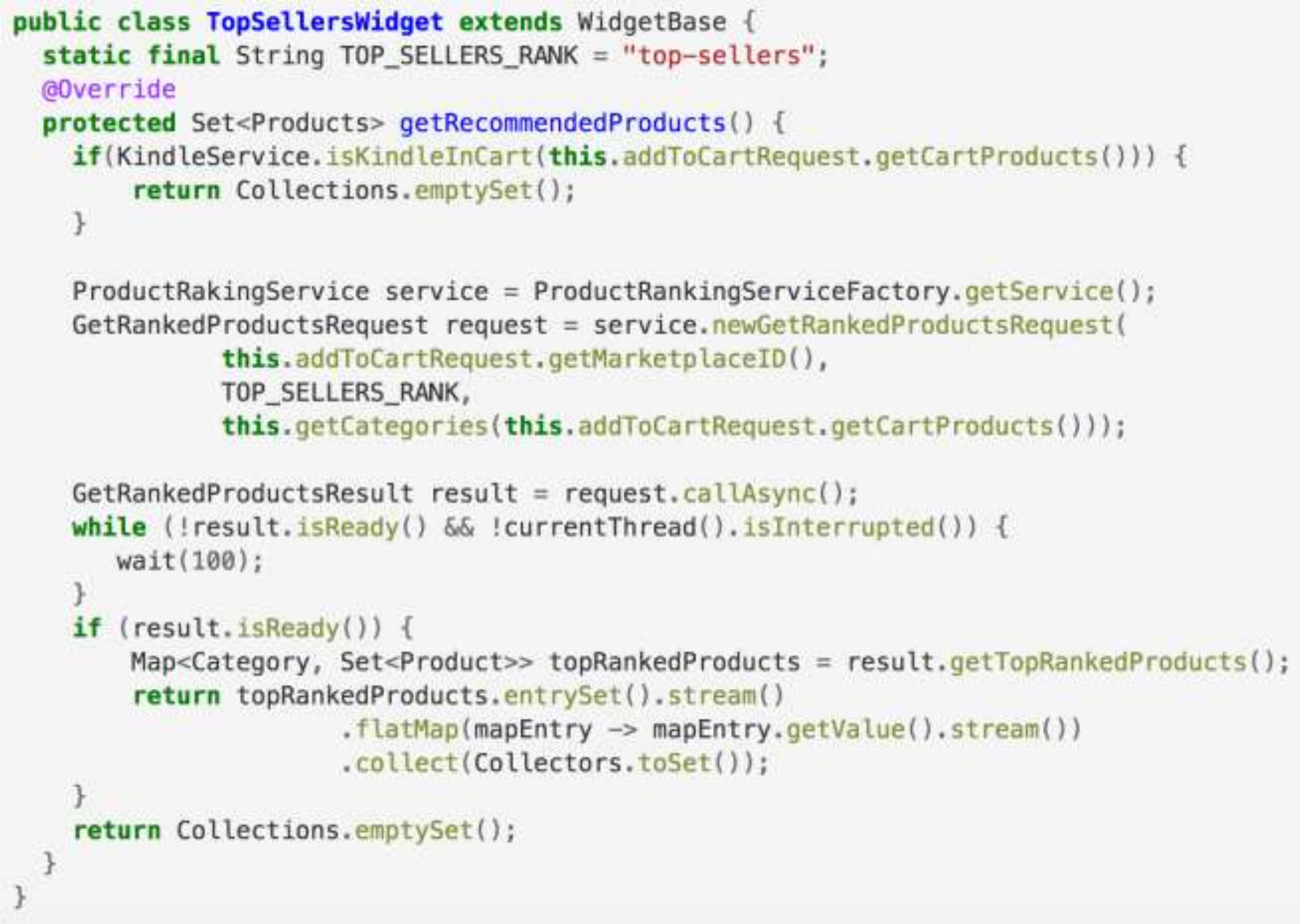
Add a 4-year warranty for Canon EOS Rebel T5 Digital SLR Camera Kit with EF-S 18-55mm... $46.59
★★★★☆ 1702

Get a $75 statement credit after first Amazon.com purchase made with new Discover it card within three months of approval*
*Terms and conditions apply. See offer for details.

Frequently Bought With *Canon EOS Rebel T5 Digital SLR Camera...*

Customers Who Shopped for *Canon EOS Rebel T5 Digital SLR Camera...* Also Shopped For

Shopping engagement

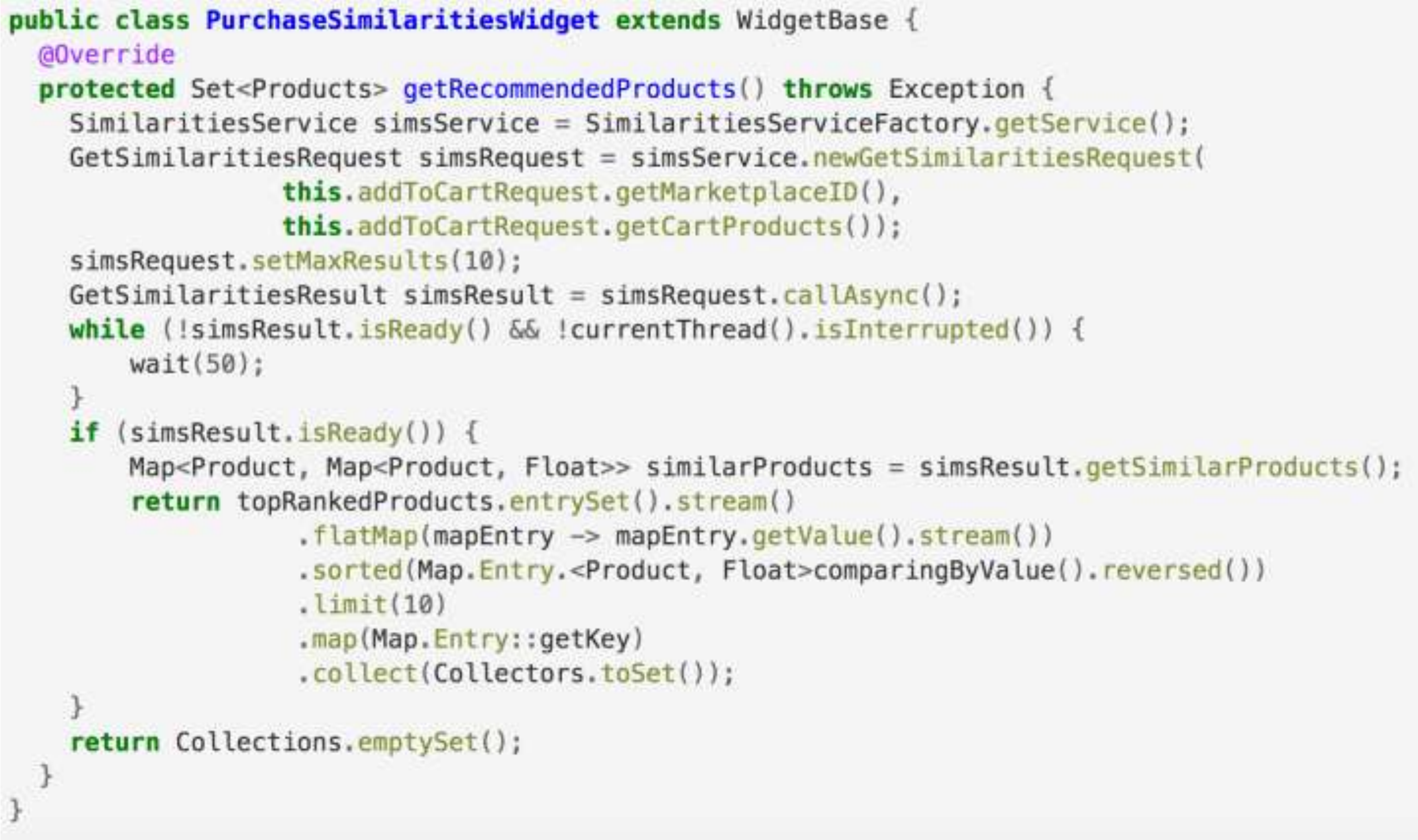


```java
public class TopSellersWidget extends WidgetBase {
  static final String TOP_SELLERS_RANK = "top-sellers";
  @Override
  protected Set<Products> getRecommendedProducts() {
    if(KindleService.isKindleInCart(this.addToCartRequest.getCartProducts())) {
        return Collections.emptySet();
    }

    ProductRakingService service = ProductRankingServiceFactory.getService();
    GetRankedProductsRequest request = service.newGetRankedProductsRequest(
              this.addToCartRequest.getMarketplaceID(),
              TOP_SELLERS_RANK,
              this.getCategories(this.addToCartRequest.getCartProducts()));

    GetRankedProductsResult result = request.callAsync();
    while (!result.isReady() && !currentThread().isInterrupted()) {
       wait(100);
    }
    if (result.isReady()) {
        Map<Category, Set<Product>> topRankedProducts = result.getTopRankedProducts();
        return topRankedProducts.entrySet().stream()
                      .flatMap(mapEntry -> mapEntry.getValue().stream())
                      .collect(Collectors.toSet());
    }
    return Collections.emptySet();
  }
}
```

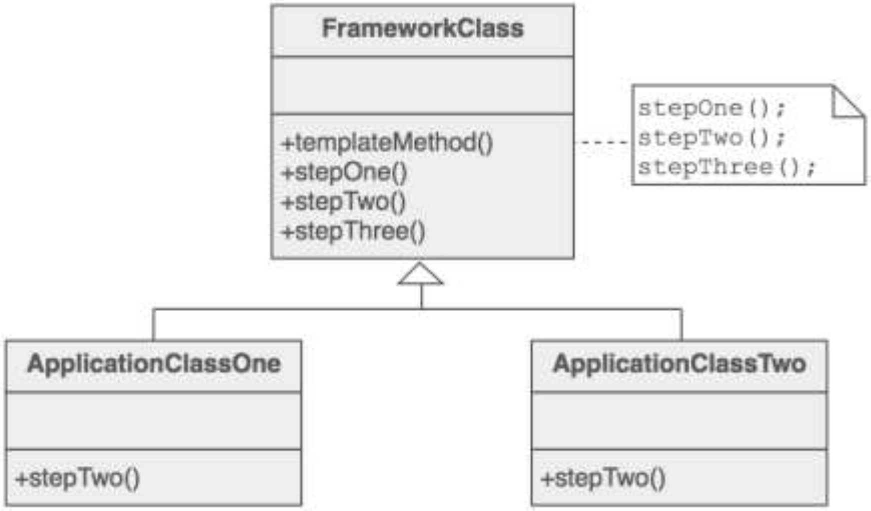
Shopping engagement



```
public class PurchaseSimilaritiesWidget extends WidgetBase {
  @Override
  protected Set<Products> getRecommendedProducts() throws Exception {
    SimilaritiesService simsService = SimilaritiesServiceFactory.getService();
    GetSimilaritiesRequest simsRequest = simsService.newGetSimilaritiesRequest(
                  this.addToCartRequest.getMarketplaceID(),
                  this.addToCartRequest.getCartProducts());
    simsRequest.setMaxResults(10);
    GetSimilaritiesResult simsResult = simsRequest.callAsync();
    while (!simsResult.isReady() && !currentThread().isInterrupted()) {
        wait(50);
    }
    if (simsResult.isReady()) {
        Map<Product, Map<Product, Float>> similarProducts = simsResult.getSimilarProducts();
        return topRankedProducts.entrySet().stream()
                  .flatMap(mapEntry -> mapEntry.getValue().stream())
                  .sorted(Map.Entry.<Product, Float>comparingByValue().reversed())
                  .limit(10)
                  .map(Map.Entry::getKey)
                  .collect(Collectors.toSet());
    }
    return Collections.emptySet();
  }
}
```

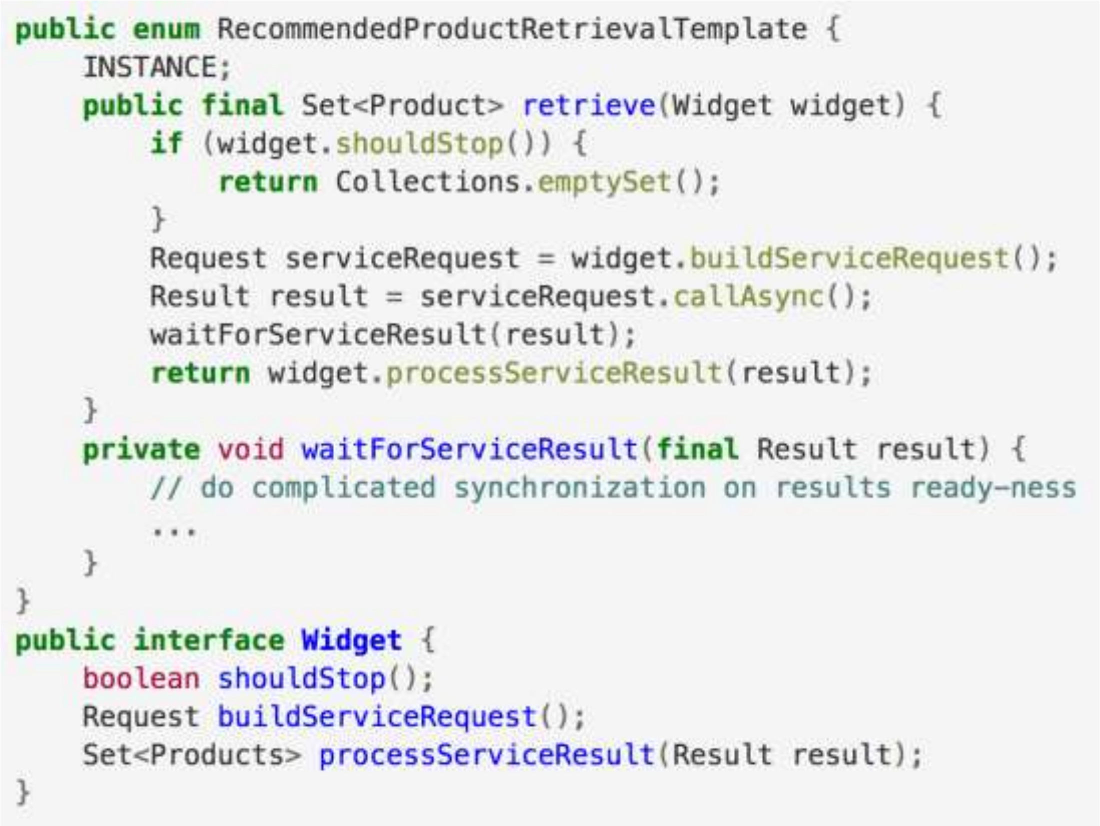
Template method

- **Problem**: two different components have significant similarities. A change common to both components implies duplicate effort.
- **Definition**: Define the skeleton of an algorithm in an operation, deferring some steps to client subclasses.
- **Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.**



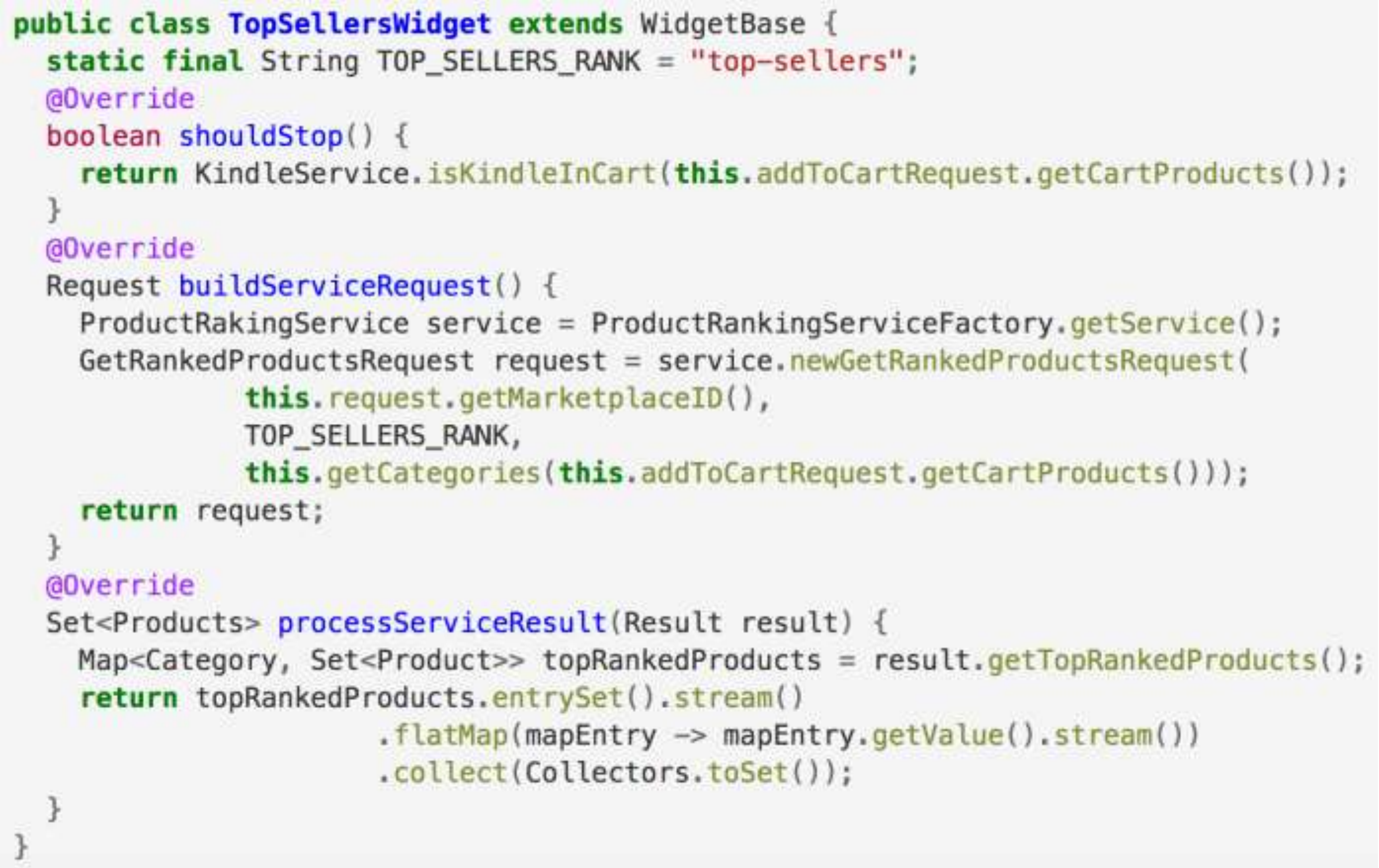
https://sourcemaking.com/design_patterns/template_method

Template method



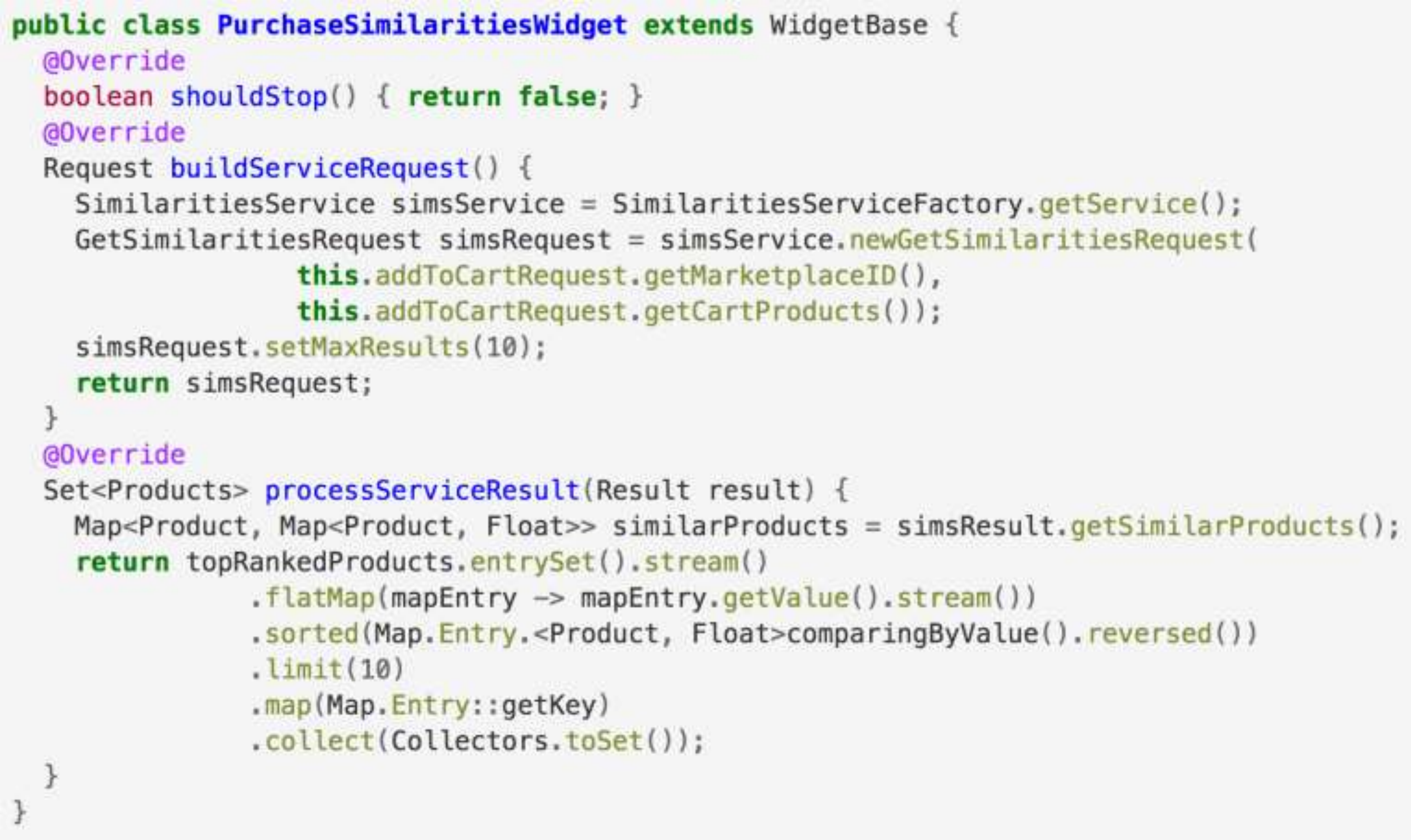
```
public enum RecommendedProductRetrievalTemplate {
    INSTANCE;
    public final Set<Product> retrieve(Widget widget) {
        if (widget.shouldStop()) {
            return Collections.emptySet();
        }
        Request serviceRequest = widget.buildServiceRequest();
        Result result = serviceRequest.callAsync();
        waitForServiceResult(result);
        return widget.processServiceResult(result);
    }
    private void waitForServiceResult(final Result result) {
        // do complicated synchronization on results ready-ness
        ...
    }
}
public interface Widget {
    boolean shouldStop();
    Request buildServiceRequest();
    Set<Products> processServiceResult(Result result);
}
```

Template method



```
public class TopSellersWidget extends WidgetBase {
  static final String TOP_SELLERS_RANK = "top-sellers";
  @Override
  boolean shouldStop() {
    return KindleService.isKindleInCart(this.addToCartRequest.getCartProducts());
  }
  @Override
  Request buildServiceRequest() {
    ProductRakingService service = ProductRankingServiceFactory.getService();
    GetRankedProductsRequest request = service.newGetRankedProductsRequest(
              this.request.getMarketplaceID(),
              TOP_SELLERS_RANK,
              this.getCategories(this.addToCartRequest.getCartProducts()));
    return request;
  }
  @Override
  Set<Products> processServiceResult(Result result) {
    Map<Category, Set<Product>> topRankedProducts = result.getTopRankedProducts();
    return topRankedProducts.entrySet().stream()
                      .flatMap(mapEntry -> mapEntry.getValue().stream())
                      .collect(Collectors.toSet());
  }
}
```

Template method



```
public class PurchaseSimilaritiesWidget extends WidgetBase {
  @Override
  boolean shouldStop() { return false; }
  @Override
  Request buildServiceRequest() {
    SimilaritiesService simsService = SimilaritiesServiceFactory.getService();
    GetSimilaritiesRequest simsRequest = simsService.newGetSimilaritiesRequest(
                  this.addToCartRequest.getMarketplaceID(),
                  this.addToCartRequest.getCartProducts());
    simsRequest.setMaxResults(10);
    return simsRequest;
  }
  @Override
  Set<Products> processServiceResult(Result result) {
    Map<Product, Map<Product, Float>> similarProducts = simsResult.getSimilarProducts();
    return topRankedProducts.entrySet().stream()
               .flatMap(mapEntry -> mapEntry.getValue().stream())
               .sorted(Map.Entry.<Product, Float>comparingByValue().reversed())
               .limit(10)
               .map(Map.Entry::getKey)
               .collect(Collectors.toSet());
  }
}
```

Template method - benefits

- All widgets know what they have to define in order to return recommendations.
- They don’t reinvent the wheel.
- They don’t duplicate async calls and results synchronization code.

Design patterns in low level programming

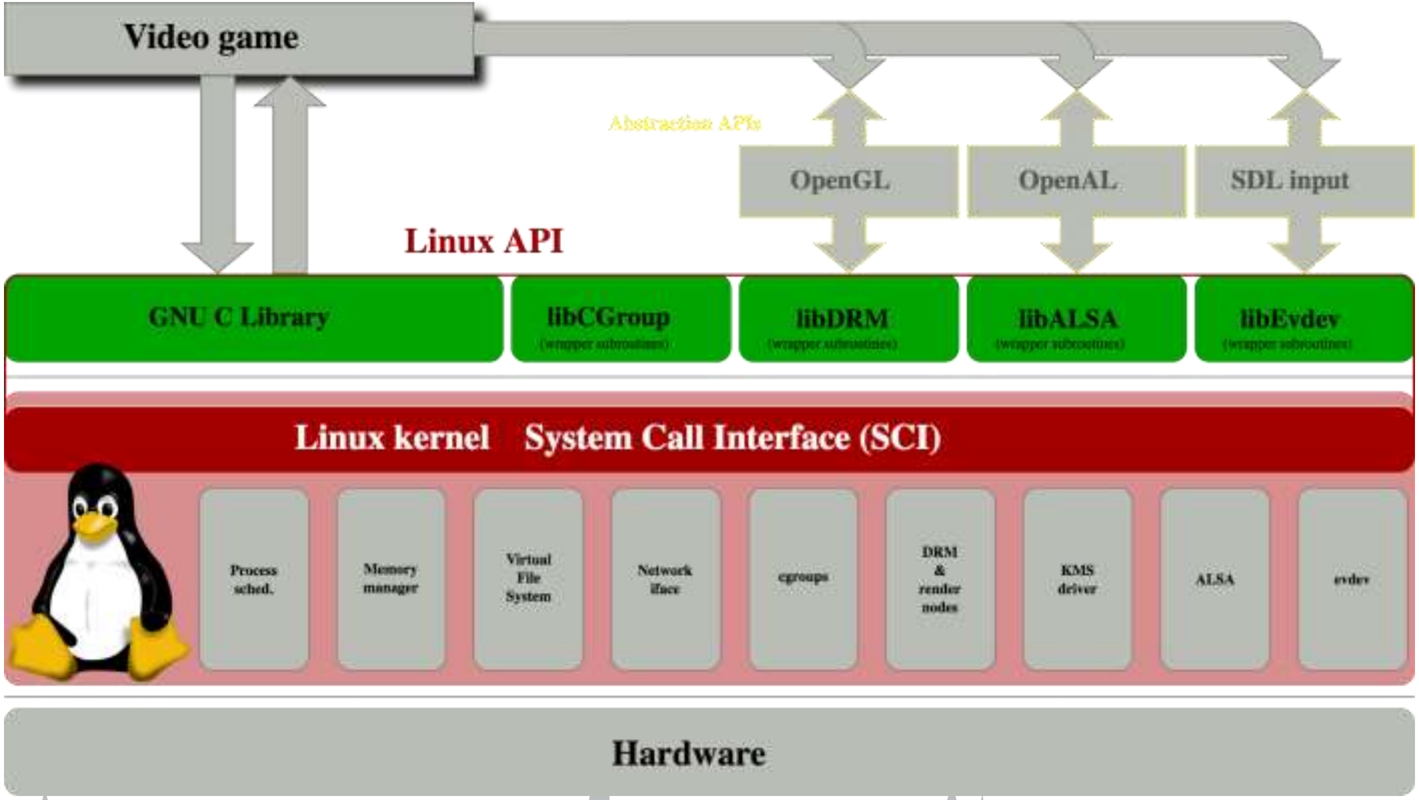
Design patterns in low level programming

- Do you trust Amazon to delegate credit card processing?
- We work hard to protect customer data

Linux Security Modules

- How access control works in kernel?
- Different security modules:
  - SELinux (NSA)
  - AppArmor (Canonical)
  - Smack (Intel)
  - TOMOYO (NTT Data Corp)
  - Yama (Canonical)

Linux Security Modules



https://upload.wikimedia.org/wikipedia/commons/4/43/Linux_API.svg

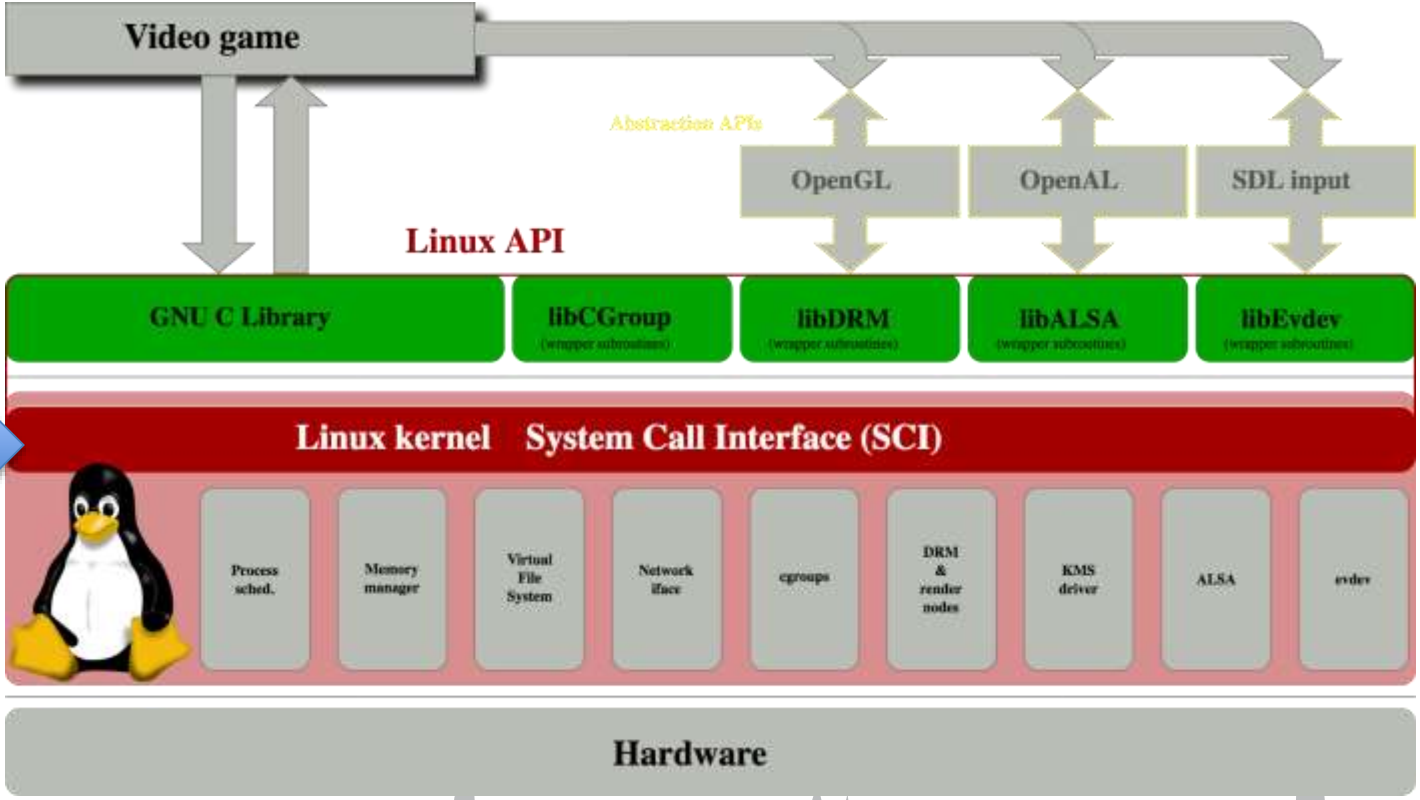
Linux Security Modules

How can one restrict syscall access?

Linux Security Modules

- In early 2000 there was no standardized kernel security framework
- NSA proposes SELinux as a monolithic solution

Linux Security Modules



2001 SELinux

https://upload.wikimedia.org/wikipedia/commons/4/43/Linux_API.svg

Linux Security Modules

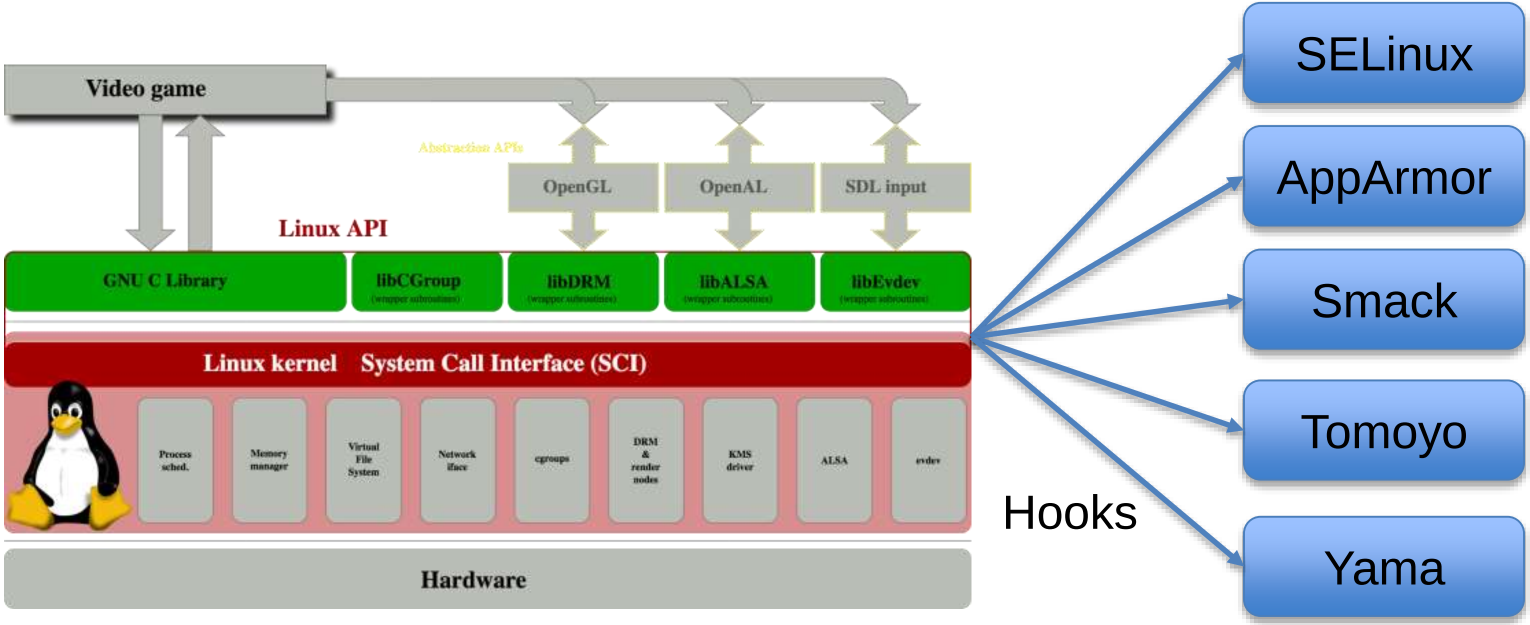
High-level pseudocode:

```
open_syscall():
    …
    selinux_access_checks()
    ...
    open_call()
```

Linux Security Modules

- In 2003 there was still the same problem
- LSM is introduced as a generic security framework
- SELinux is approved in the mainline

Linux Security Modules



Hooks

SELinux

AppArmor

Smack

Tomoyo

Yama

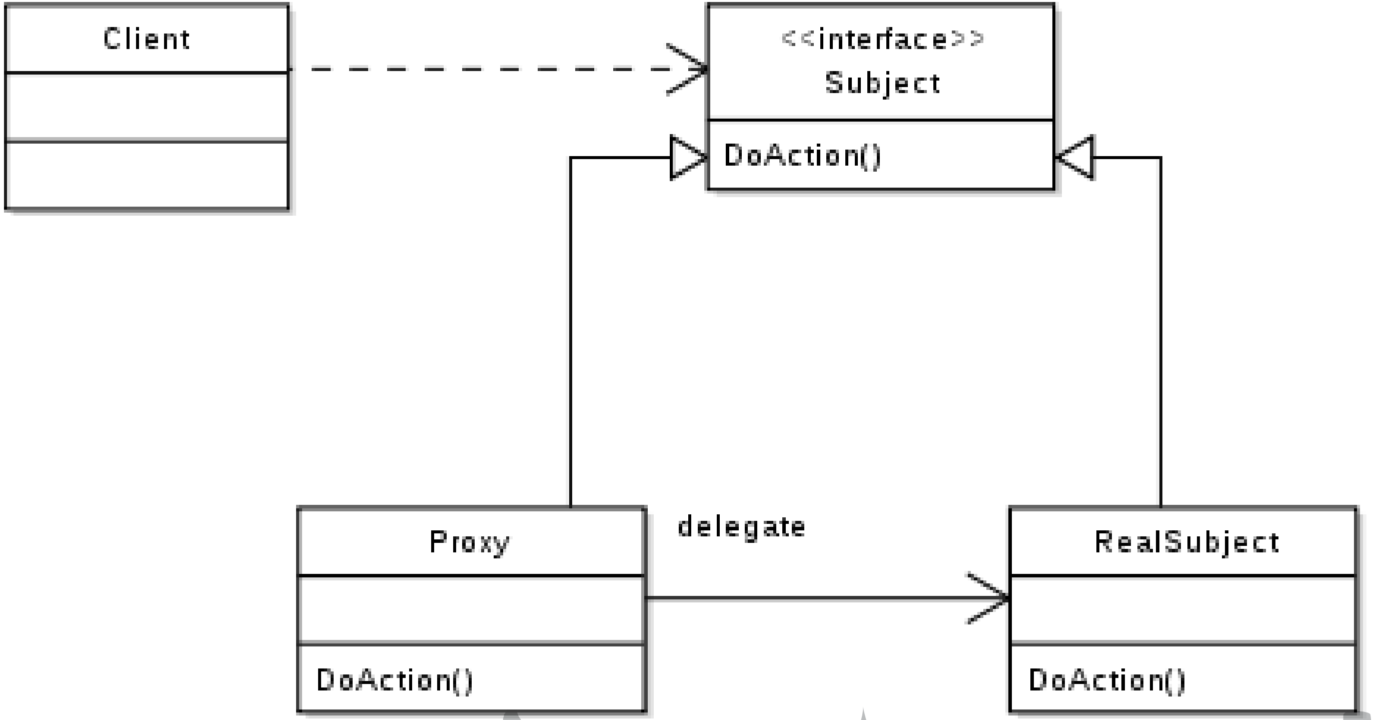
https://upload.wikimedia.org/wikipedia/commons/4/43/Linux_API.svg

Linux Security Modules

High-level pseudocode:

```
register_selinux_hooks()
open_syscall():
    …
    process_generic_hooks()
    ...
    open_call()
```

Linux Security Modules – Proxy Pattern



https://upload.wikimedia.org/wikipedia/commons/thumb/7/75/Proxy_pattern_diagram.svg/800px-Proxy_pattern_diagram.svg.png

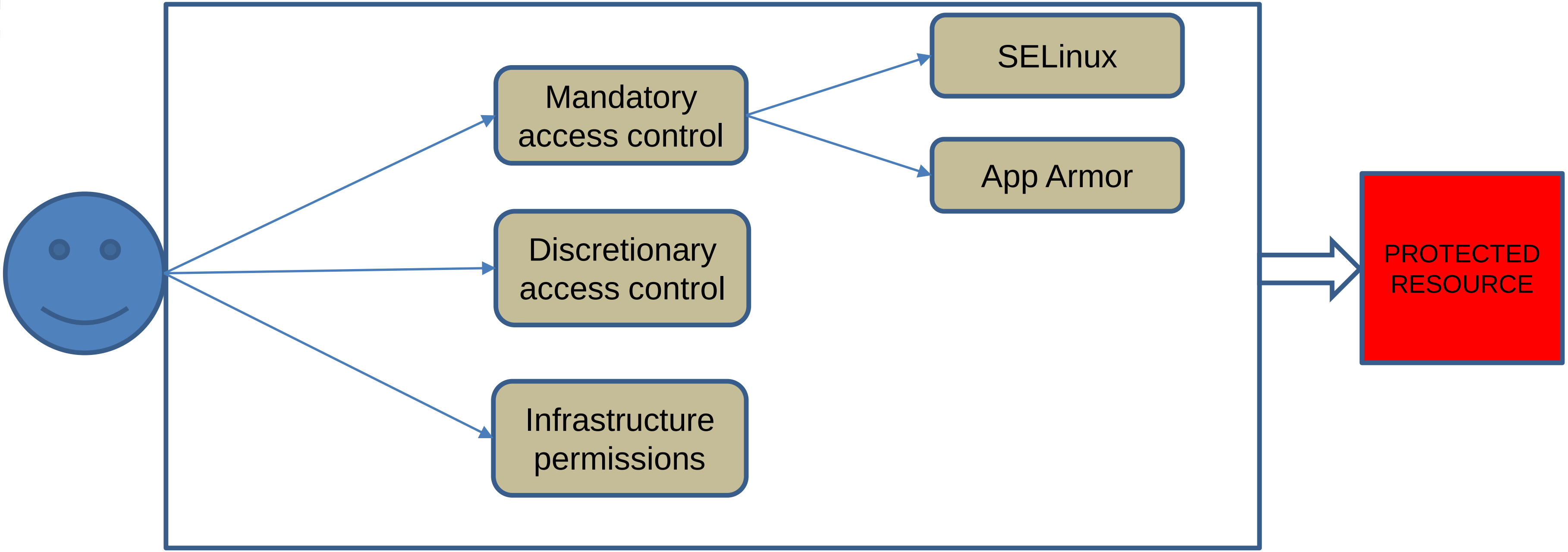
Linux Security Modules – Proxy Pattern

- Rejecting initial SELinux proposal was a chance to build generic frameworks
- Proxy pattern adds an indirection layer
- Reduces complexity from critical components
- Adds a plus on availability since critical code is not modified often

Managing permissions

- Default Discretionary Access Control(DAC) in Linux has: read, write and execute
- Sometimes it is not enough, you want additional granularity (heap dump, stack trace)
- Existing frameworks might be too complex with a steep learning curve
- There are multiple kinds of permissions for each resource
- Sometimes you must combine different technologies to achieve a goal

Managing permissions



Managing permissions

High-level pseudocode

```
init_mac_system()
init_dac_system()
init_infrastructure_permissions()
…
check_mac_access(user, resource, action)
check_dack_access(user, resource, action)
check_infrastructure_permissions(user, resource, action)
```

Managing permissions - Facade

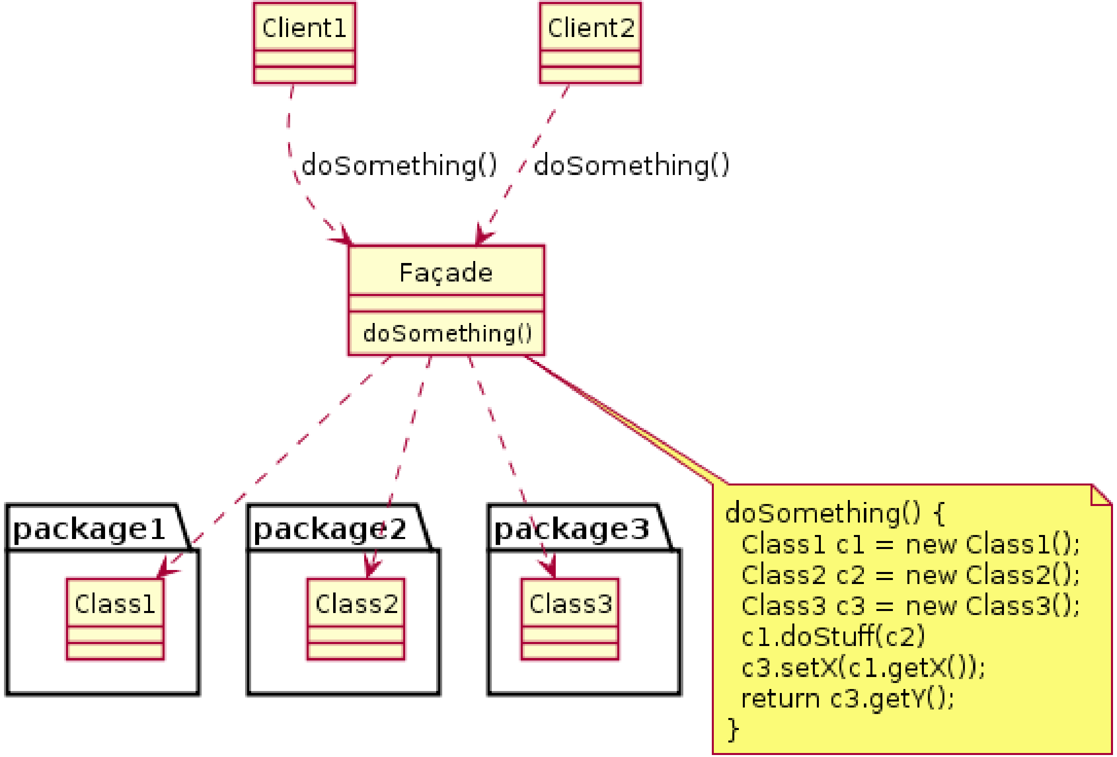
How can we make it simpler?

Managing permissions - Facade

High-level pseudocode:

```
init_permission_system()
…
check_permissions(user, resource, action)
```

Managing permissions - Facade



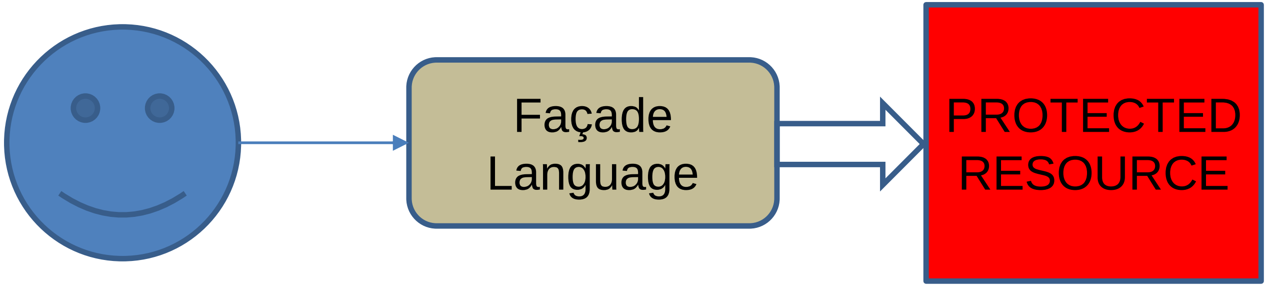
```
doSomething() {
  Class1 c1 = new Class1();
  Class2 c2 = new Class2();
  Class3 c3 = new Class3();
  c1.doStuff(c2)
  c3.setX(c1.getX());
  return c3.getY();
}
```

https://en.wikipedia.org/wiki/Facade_pattern#/media/File:Example_of_Facade_design_pattern_in_UML.png

Managing permissions - Facade

- Provide a unified interface to a set of interfaces in a subsystem
- Transform a complicated subsystem by using a simpler interface

Managing permissions - Facade



Security work is never ending story.

Follow design patterns to make it easier to understand and maintain.

A project’s lifecycle

A project's lifecycle

http://www.ign.com/articles/2015/10/05/this-life-size-lego-batmobile-is-unbelievable

A project’s lifecycle

- Handover
- Requirements
- Design
- Develop & Test
- Maintenance

The Vision

The Result

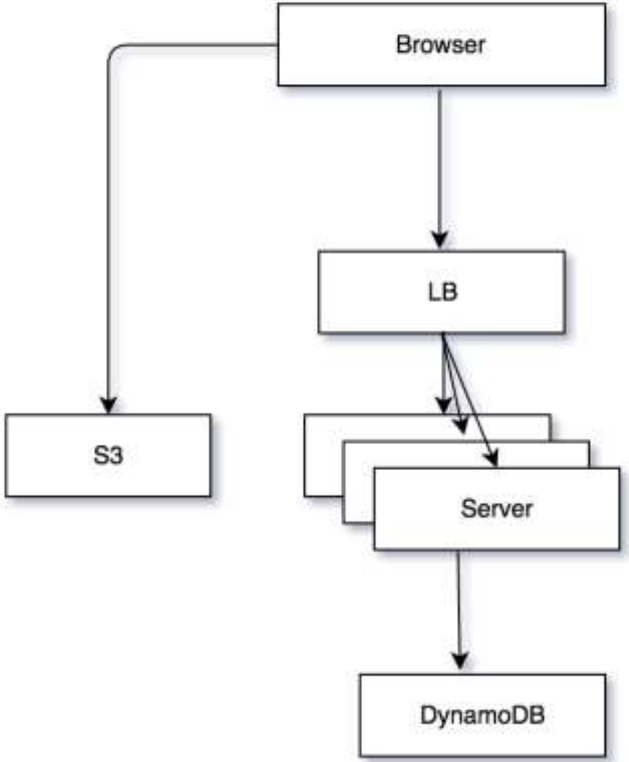
© barryoreilly.com

Handover

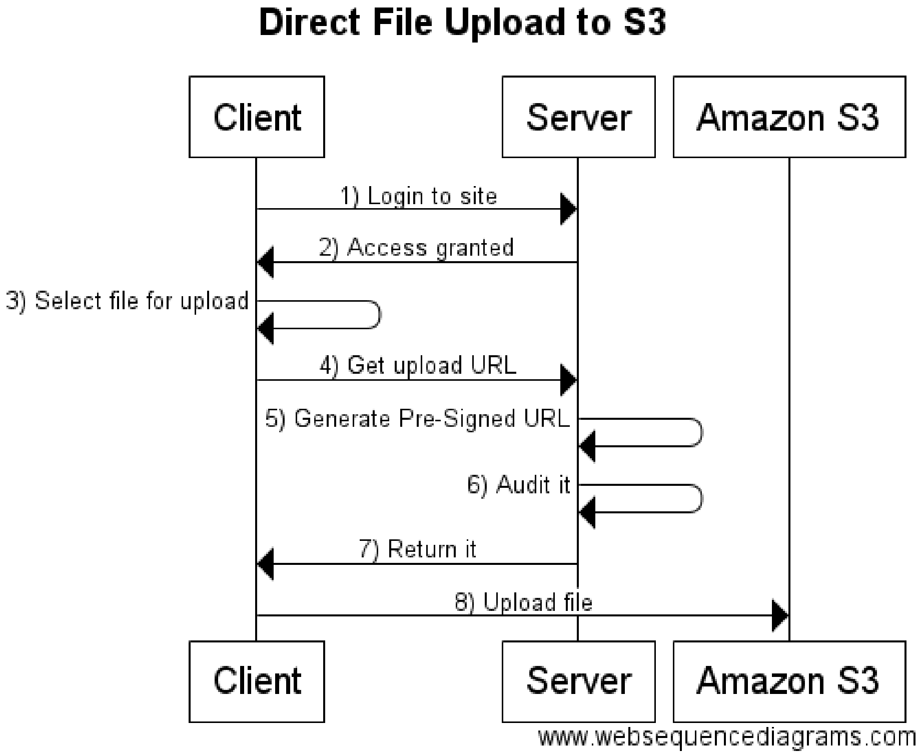
- **The project**
  - How big is the project
  - How many users does the system have
- **The communication**
  - Reverse engineering
  - You are given a presentation
  - Q/A on conference call
  - Emails

Handover

- Documentation
  - Architecture diagram
  - Database diagram
  - Sequence diagram
  - Deployment document



Handover


Direct File Upload to S3
Client
Server
Amazon S3
1) Login to site
2) Access granted
3) Select file for upload
4) Get upload URL
5) Generate Pre-Signed URL
6) Audit it
7) Return it
8) Upload file
Client
Server
Amazon S3
www.websequencediagrams.com

Requirements

- A list of business requirements
  - Forward engineering
  - Small requirements or lots of new functionalities
- More requirements from developers
  - Amazon’s working backwards philosophy
  - Developer in the driver seat
  - Start with the customer
  - User research
  - Discover new customers
- Innovation
- Review the list of requirements with peers and stakeholders

Design

- **Steps for a successful design**
  - Create an architecture overview document
  - Create a prototype
  - Create final design document
  - Validate the design with engineering leaders in the company

Design

- **Things to consider**
- System integration
  - The other systems it needs to communicate with
  - Message bus? Caching layer? Search? Big data?
  - The scale at which it will operate
  - Create a new module / service or build on existing stack?
    - More services / less services
    - SOA at Amazon
- Technologies
  - Languages and frameworks to use
  - Should it use an existing library (Open source, AWS)?

Amazon Rekognition

AWS Lambda

redis

Spark

Develop, develop, develop

- Handover
- Requirements
- Design
- **Develop & Test**

- Is this all?

Maintenance

- Define environments
  - Pre-production and production stages
  - Host infrastructure
- Monitoring
  - Performance metrics
  - Health metrics
- Alarms
- Optimizations

Nagios®

AWS CloudWatch

Thank you!

Please help us improve by filling out the survey for the presentation.